目录

[1. 任务概述 2](#_Toc413386307)

[1.1 目标 2](#_Toc1370438183)

[1.2 开发环境 2](#_Toc1492088517)

[1.3 需求概述 2](#_Toc560688649)

[1.4 条件和限制 3](#_Toc851178112)

[2. 总体设计 3](#_Toc1257050333)

[2.1 程序流程 3](#_Toc1828379958)

[2.2 组件功能 4](#_Toc906332648)

[2.3 数据结构(功能说明) 4](#_Toc260937223)

[3. 接口设计 5](#_Toc545015203)

[3.1 内部接口(组件互相调用或主程序调用) 5](#_Toc1972361016)

[3.2 外部接口 6](#_Toc311634652)

# 任务概述

## 目标

yaml是一种数据序列化语言，其语法简洁直观，通过空格缩进表示层级结构，在数据结构表达方面有较大优势，常用于编写配置文件。

该项目的最终目标为实现一个C++的yaml库，使用递归下降分析法对yaml语言进行解析，实现对yaml的读写功能。

## 开发环境

* OS: Ubuntu-20.04
* Editor: vscode
* Compiler: gcc-9.4.0
* Build: cmake-3.16.3
* VCS: git

## 需求概述

* 实现yaml相关类型：null、object、array、int、real、string，且string类型实现保留换行和折叠换行
* 通过std::istream输入流读取yaml源数据
* 通过std::ostream输出流将yaml类型输出
* 根据yaml语法规则，使用递归下降子程序法实现yaml解析

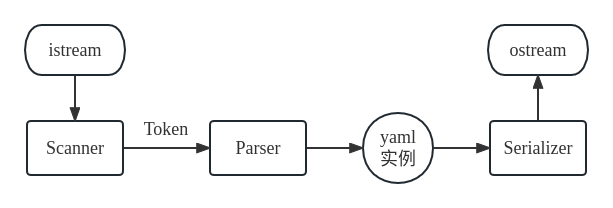
## 1.4 条件和限制

* 标准: c++17
* 库: STL
* 禁止使用任何第三方库

# 总体设计

## 程序流程

对外部调用者，yaml库主要提供了yaml类型的实现和yaml的读写接口。而yaml库内部则通过Scanner、Parser、Serializer三个组件实现相关的功能



## 组件功能

该yaml库主要由以下组件组成

* Scanner: 通过istream接收输入流，从头到尾按顺序扫描源文本，进行词法分析，将文本转换为一组token序列
* Parser: 接收来自Scanner生成的token，在token的基础上进行语法分析，检测语法正确性并生成相应的yaml实例
* Serializer：接收yaml实例，将该实例进行序列化并通过ostream将序列化后的内容输出到制定位置

## 2.3 数据结构(功能说明)

Token: 词法分析解析结果

type: token类型枚举值

value: token的原始字面量

Value: yaml类型基类

type: Value类型枚举值

serialize(): yaml类型序列化虚函数

Null\_Value: yaml null类型

Object\_Value: yaml object类型

Array\_Value: yaml Array类型

Int\_Value: yaml int类型

Real\_Value: yaml real类型

String\_Value: yaml string类型

# 3. 接口设计

## 3.1 内部接口(组件互相调用或主程序调用)

Scanner:

1. nextToken()

功能：获取下一个token

参数：无

返回值：一个Token(Token)

1. lineno()

功能：获取当前行号，用于报告错误

参数：无

返回值：当前行号(unsigned int)

1. colno()

功能：获取当前列号，用于报告错误

参数：无

返回值：当前列号(unsigned int)

Parser:

1. parse()

功能：开始解析

参数：无

返回值：无

## 3.2 外部接口

Serializer:

1. Serialize(Value &, ostream &):

功能：将yaml类型对象序列化输出至给定ostream

参数1：Value引用

参数2：ostream引用

返回值：无

1. Serialize(Value &):

功能：将yaml类型对象序列化为字符串

参数：Value引用

返回值：序列化后的字符串(std::string)